netcat und *socat* – vielseitige Werkzeuge fürs Netz

Kontaktmittel

Michael Plura

Gelegentlich muss sich ein Systemverwalter zu Fuß in sein Netz begeben. Hier helfen *netcat* und sein Nachfolger *socat*.

Mit *netcat*, dem *cat* fürs Netz, steht dem Administrator ein universelles Werkzeug zur Datenübertragung über TCP oder UDP zur Verfügung. Es ist einfach gehalten und verarbeitet beliebige Datenströme der Standardein- und -ausgabe. *netcat* kann als Server oder Client fungieren.

Klassische Unixe bringen *netcat* als *nc* bereits in ihrer Basisinstallation mit. Die BSD-Varianten, Fedora und openSUSE setzen auf die OpenBSD-Neufassung. Sie beherrscht IPv6, Proxies und Unix-Sockets. Debian und Ubuntu kennen gleich drei: die als traditionell bezeichnete GNU-Neufassung, die jüngere *netcat6*- oder *nc6*-Überarbeitung mit IPv6- und erweitertem UDP-Support und die besagte OpenBSD-Variante.

Deshalb ist *netcat* ins Debian-Umbausystem *alternatives* eingebunden. Von */bin* zeigen zwei Softlinks mit Namen *nc* und *netcat* auf gleichnamige Softlinks in */etc/alternatives*. Die wiederum verweisen bei Debian auf das standardmäßig installierte */bin/nc.traditional*, bei Ubuntu auf */bin/nc.openbsd*. Wer die Version wechseln will, installiert mit *apt-getinstall <netcat6|netcat-openbsd|netcat-traditional>* das nötige Paket, gibt *update-alternatives --config nc* ein und trifft seine Auswahl. Das verbiegt die zugehörigen Softlinks auf das gewünschte Binary und das passende Manual.

Komplikationen können sich durch die unterschiedlichen Parameter ergeben. Den von Ballast befreiten Versionen *nc6*- und *nc.openbsd* fehlen die gefährlichen Optionen *–c* und *–e*, die einen String oder eine Datei nach dem Verbindungsaufbau ausführen, ebenso wie *–o* für den Hexdump des Traffic. Beides lässt sich mit einem Skript oder *tcpdump* besser erledigen.

Nervig ist der Unterschied bei der Angabe des Ports mit dem Schalter *–p*: *nc.traditional* und *nc6* benötigen die Angabe zur Definition des lokalen Ports, an dem es lauschen soll, OpenBSDs *nc* hingegen legt damit den Source-Port für Remote-Verbindungen fest.

Einfach ist zumindest die Funktionsweise: Um die Standardein- und -ausgaben zweier Terminals miteinander zu verbinden, weist man die OpenBSD-Version mit *nc –l <port>* und die anderen mit *nc –l –p <port>* auf einem der Systeme an, an einem Port zu lauschen. Das zweite System verbindet sich mit dem ersten nach Angabe eben jenes Rechners und des geöffneten Ports mit *nc <host> <port>*.

Ersatz für Spezialwerkzeug

Ähnlich lassen sich Dateien unverschlüsselt übertragen, wenn beispielsweise kein *scp* verfügbar ist. Dazu öffnet man auf dem Empfänger das Tor etwa mit *nc –l 4000 | tar xvzf –* und sendet das aktuelle Verzeichnis mit dem Befehl *tar –cvzf – * | nc <zielrechner> 4000*. Das sendende *netcat* beendet man mit Strg + C.

In der Form *nc –zv <host> 1–1024* kann *netcat* einfache Port-Scans durchführen. Wie mit *telnet*, aber universeller, prüft *netcat* auch Dienste, beispielsweise einen Webserver mit *printf 'HEAD / HTTP/1.0\r\n\r\n' | nc <host> 80*. *printf* funktioniert hier besser als *echo*, weil Letzteres die *\r\n*-Kombination nicht wie gewünscht handhabt. Im Netz kursieren Anleitungen, wie man mit *nc* einen Mini-Webserver aufsetzt, der in einer Endlosschleife eine *index.html* an Port 80 ausliefert – mit modernen Browsern funktioniert das aber nicht.

Hier bietet sich stattdessen *socat* an, das mächtige *cat* für Sockets, das als eine Art universelles Relay fungieren kann. Damit lassen sich zwei bidirektionale Datenströme verbinden, wobei *socat* mit Dateien, Pipes, Geräten wie Terminals und Modems oder eben Sockets arbeitet – gleichgültig, ob raw, IPv4, IPv6, UDP/TCP oder SSL. Allerdings muss man *socat* in der Regel nachinstallieren.

Wie *telnet* oder *nc* kann sich auch *socat* mit einem Dienst verbinden, etwa durch *socat – TCP4:<host>:80* mit dem Webserver. Das Minuszeichen kennzeichnet den Datenstream von *STDIN* zu *STDOUT*, die folgenden Parameter definieren das Protokoll, das Zielsystem und den Port. Die Eingabe von *HEAD / HTTP/1.0* und eine Leerzeile haben denselben Effekt wie im letzten *nc*-Beispiel.

Der Befehl *socat TCP4-LISTEN:www, reuseaddr,fork TCP4:<host>:www* erzeugt einen Port-Forwarder, der den eingehenden Web-Traffic auf *<host>* umleitet. Das Argument *fork* veranlasst *socat*, die Verbindung einem Kindprozess zu überlassen und auf weitere zu horchen.

Umgekehrt lässt sich mit *socat OPENSSL-LISTEN:443,reuseaddr,pf=ip4,fork, cert=server.pem,cafile=client.crt TCP4-CONNECT:localhost:80* einem Dienst, der keine verschlüsselten Zugriffe verarbeiten kann, eine Verschlüsselung vorschalten. *socat* nimmt die Zugriffe auf Port 443 an, überprüft Clients anhand der Zertifikate in der *client.crt* und leitet die Anfrage an den lokalen Port 80 weiter.

Auch eine fehlende SSH kann *socat* ersetzen. Passende Zertifikate vorausgesetzt, lauscht der Server nach Eingabe von *socat OPENSSL-LISTEN:44443, reuseaddr,fork,cert=server.pem,cafile=client.crt,verify=1 exec'<befehl>'* auf SSL-Verbindungen. Der Client greift mit *socat –OPENSSL:<server>:44443,cert=client.pem,cafile=server.crt* über SSL auf den Server zu, führt *<befehl>* aus und bekommt dessen Ausgabe zurück.

netcat und *socat* eignen sich gleich einem Schweizer Taschenmesser zum schnellen Analysieren und Beheben kleinerer Macken. Allein das Manual von *socat* offenbart das beachtliche Potenzial des zwar in die Jahre gekommenen, aber praktischen Netzwerkzeugs. (sun)

Michael Plura

lebt in Schweden und ist freier Autor mit den Schwerpunkten IT-Sicherheit, Virtualisierung und freie Betriebssysteme.

Alle Links: www.ix.de/ix1711134